Lecturer: Prof. Dr. Henning Sprekeler

Assistant: Simone Ciceri
(simone.ciceri@charite.de)

Homework: Deep Linear Neural Networks

April 16, 2024

The solutions for these exercises should be handed in before **April 26, 2024 at 10:15 am** as a pdf through the Moodle interface. Please post your questions regarding the exercise on the Moodle forum.

1 Introduction

In this exercise we will analyze how a linear neural network learns the structure of its training data. The network maps inputs $\mathbf{x} \in \mathbb{R}^{N_1}$ to outputs $\mathbf{y} = W^2 W^1 \mathbf{x} \in \mathbb{R}^{N_3}$ via weight matrices W^2 and W^1 of size $N_3 \times N_2$ and $N_2 \times N_1$ (Figure 1, left). The weights are optimized using gradient descent on the mean squared error

$$E = \frac{1}{2n} \sum_{i=1}^{n} \|\mathbf{y}^i - W^2 W^1 \mathbf{x}^i\|^2, \tag{1}$$

where the sum runs over all n training samples. Without nonlinear activation functions, the network can only learn linear transformations of its input. But the objective E is still a nonlinear function of the weights, leading to some of the interesting phenomena observed when training nonlinear networks.

A key challenge in analyzing deep networks is that the contribution of a given weight to the network's output depends on many other weights. We will tackle this challenge by decomposing the network into independent chains that map input modes (combinations of weights) to corresponding output modes. Each of these chains can be analyzed separately, simplifying the analysis.

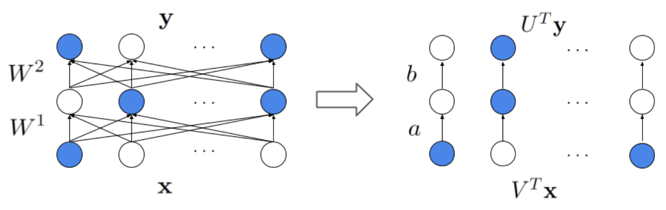

Figure 1: Left: Linear network in which each input dimension activates multiple network units. Right: Strategy for analyzing deep linear networks—a change of variables and orthogonal weight initialization.

2 Exercises

1. Gradient descent on E yields the update rule

$$\Delta W^1 := -\eta \frac{\partial E}{\partial W^1} = \eta {W^2}^T (\Sigma^{yx} - W^2 W^1 \Sigma^{xx}), \tag{2}$$

   for the input-to-hidden weight W^1. Here η is the learning rate, $\Sigma^{xx} = \frac{1}{n} \sum_i \mathbf{x}^i {\mathbf{x}^i}^T$ is the input covariance matrix and $\Sigma^{yx} = \frac{1}{n} \sum_i \mathbf{y}^i {\mathbf{x}^i}^T$ is the input-output covariance matrix. Derive the analogous rule ΔW^2 for the hidden-to-output weights.

For small learning rates, the parameter update (2) approximates the continuous time dynamics

$$\tau \frac{dW^1}{dt} = {W^2}^T (\Sigma^{yx} - W^2 W^1 \Sigma^{xx}), \tag{3}$$

where $\tau = 1/\eta$ is a time constant, and $\frac{dW^2}{dt}$ follows analogous dynamics. The difficulty in analyzing these equations is that they are coupled: The change of a parameter depends not only on its own value but also on that of the others (Figure 1, left). We will overcome this difficulty by decomposing the network into independent chains, decoupling the update equations (Figure 1, right). The first step is to apply a change of variables using the singular value decomposition (SVD) of the input-output covariance matrix:

$$\Sigma^{yx} = USV^T = \sum_{i=1}^{r} s_i \mathbf{u}^i {\mathbf{v}^i}^T, \tag{4}$$

with r the rank of Σ^{yx}; V and U are orthonormal matrices ($V^T V = I_{N_1}$, $U^T U = I_{N_3}$) whose columns $\mathbf{u}^i, \mathbf{v}^i$ correspond to independent modes of variation in the input and output, respectively; S is a diagonal matrix containing the ordered singular values $s_1 \geq s_2 \geq \ldots \geq s_r \geq 0$ which characterize "correlation strengths" between input and output modes.

2. Performing the change of basis $\overline{W^1} = W^1 V$, $\overline{W^2} = U^T W^2$ and assuming that the inputs are uncorrelated ($\Sigma^{xx} = I$), equation (3) simplifies to:

$$\tau \frac{d}{dt} \overline{W}^1 = {\overline{W}^2}^T (S - \overline{W}^2 \overline{W}^1), \tag{5}$$

   Derive $\tau \frac{d}{dt} \overline{W}^2$ from ΔW^2.

3. The i-th column $\mathbf{a}^i$ of $\overline{W^1}$ contains the weights from input mode $\mathbf{v}^i$ to the hidden layer; the ith row ${\mathbf{b}^i}^T$ of $\overline{W^2}$ contains the weights from the hidden layer to the ith output mode $\mathbf{u}^i$. They change according to :

$$\tau \frac{d}{dt} \mathbf{a}^i = (s_i - \mathbf{a}^i \cdot \mathbf{b}^i) \mathbf{b}^i - \sum_{j \neq i} (\mathbf{a}^i \cdot \mathbf{b}^j) \mathbf{b}^j, \tag{6}$$

$$\tau \frac{d}{dt} \mathbf{b}^i = (s_i - \mathbf{a}^i \cdot \mathbf{b}^i) \mathbf{a}^i - \sum_{j \neq i} (\mathbf{a}^j \cdot \mathbf{b}^i) \mathbf{a}^j. \tag{7}$$

   Here $\cdot$ denotes the inner product between two vectors. Derive equation (7). Tip: Don't write out the individual scalar terms resulting from the matrix products. Instead, sketch the layout of the vectors $\mathbf{a}^i$ and $\mathbf{b}^i$ in the expression for $\tau \frac{d}{dt} \overline{W}^2$.

4. Show that the equations (6) & (7) arise from gradient descent on the function

$$F = \frac{1}{2\tau}\sum_i (s_i - \mathbf{a}^i \cdot \mathbf{b}^i)^2 + \frac{1}{2\tau}\sum_{j\neq i}(\mathbf{a}^i \cdot \mathbf{b}^j)^2. \tag{8}$$

   Interpret the two terms of F: How would minimizing each term individually change the parameters?

The second terms on the right-hand sides of (6) and (7) couple the equations for $i \neq j$. Assume that the connectivity vectors are aligned within mode and orthogonal between modes, i.e. $\mathbf{a}^i = a_i\mathbf{r}^i$ and $\mathbf{b}^i = b_i\mathbf{r}^i$ for orthonormal vectors $\mathbf{r}^i$ and scalars a_i, b_i. (This assumption will hold for the orthogonal initialization $W^1 = RD^1V^T, W^2 = U^TD^2R$, with D^1 and D^2 are diagonal and R orthonormal.)

5. Explain that orthogonality reduces (6) & (7) to a system of r decoupled 2-dimensional equations:

$$\tau\frac{d}{dt}a_i = (s_i - a_ib_i)b_i \tag{9}$$

$$\tau\frac{d}{dt}b_i = (s_i - a_ib_i)a_i. \tag{10}$$

6. The 2-dimensional system can be easily be solved under the assumption that the two modes are equal ($a = b$); you will investigate this solution in the programming exercise. Here, however, we will try to get an intuitive understanding of the system's behavior without making further assumptions.
   a) Compute the system's fixed points (a^*, b^*).
   b) Linearize the system around the fixed points.
   c) Use the linearized system's eigenvalues to determine each fixed point's type (e.g., center, saddle etc.).
   d) Sketch the phase portrait of the system, along with sample trajectories. Where will the system end up after many parameter updates?

3 References and further reading

The analysis in this exercise is based on the work from Saxe et al. [5] which also proposed the orthogonal initialization from exercise 5. Deep linear networks have also been used to investigate why –contrary to conventional wisdom– overfitting [1] and non-convexity [2] don't necessarily pose problems for deep networks. Beyond their importance as tractable models [4], deep linear networks also have practical utility: They outperform established models as recommender systems [3].

References

[1] Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.

[2] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. *arXiv preprint arXiv:1810.02281*, 2018.

[3] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems*, pages 7411–7422, 2019.

[4] Andrew Saxe, Stephanie Nelli, and Christopher Summerfield. If deep learning is the answer, then what is the question? *arXiv preprint arXiv:2004.07580*, 2020.

[5] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.